



Kyle B. Rinehart  
Klarquist Sparkman et al  
121 SW Salmon Street  
Suite 1600  
Portland, Oregon 97204  
Telephone: 503/226-7391

Our Ref. No. 3382-67641  
For: MULTI-LAYER RUN LEVEL  
ENCODING AND DECODING  
Inventors: Liang et al.  
Express Mail No. EV331580878US  
Mailed: April 15, 2004

Figure 1

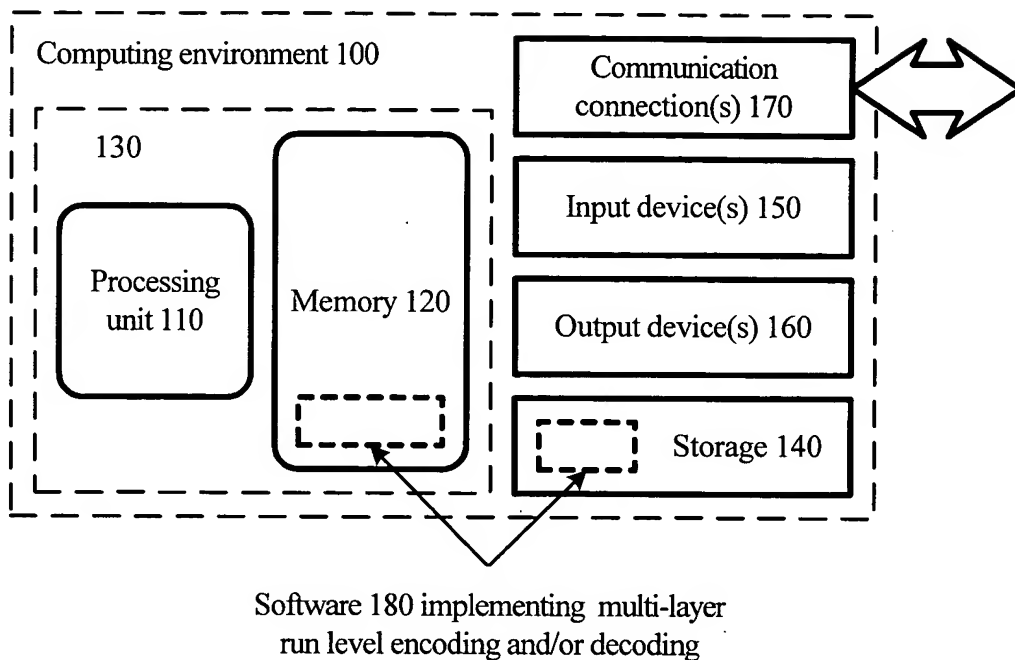


Figure 4

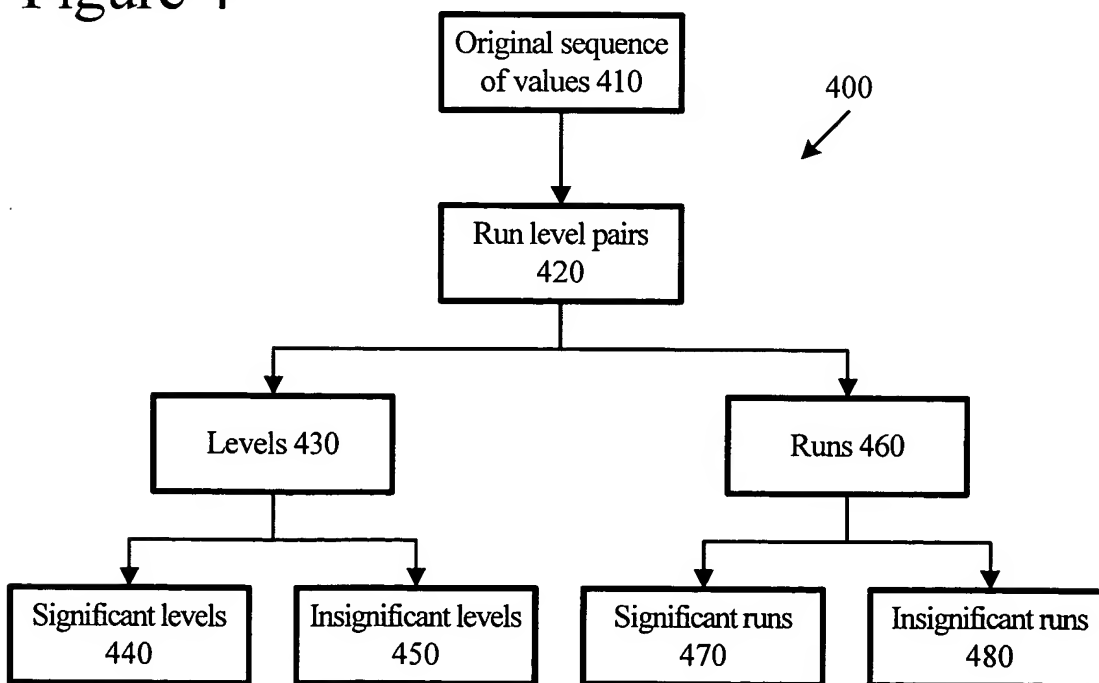


Figure 2

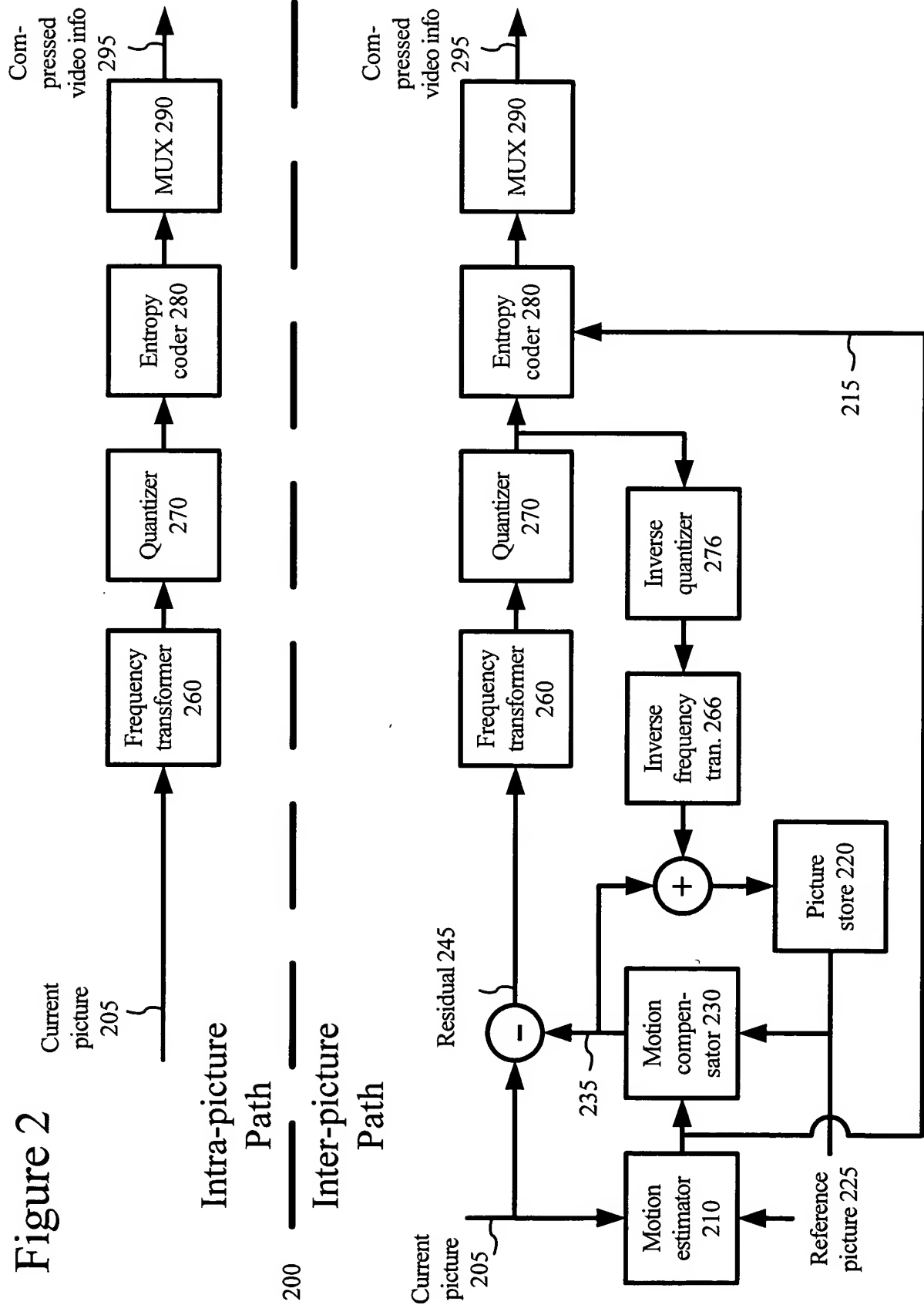


Figure 3

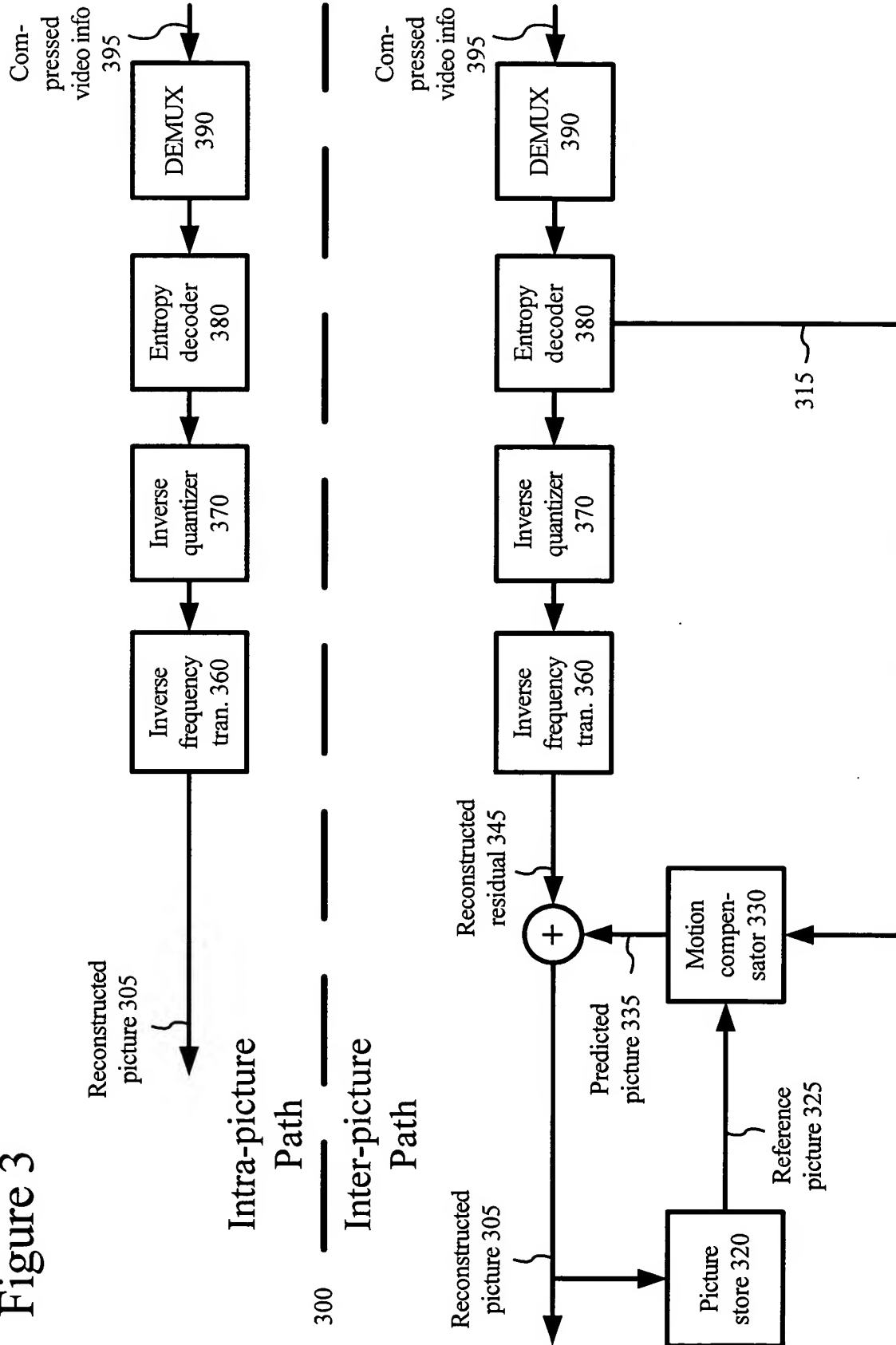


Figure 5

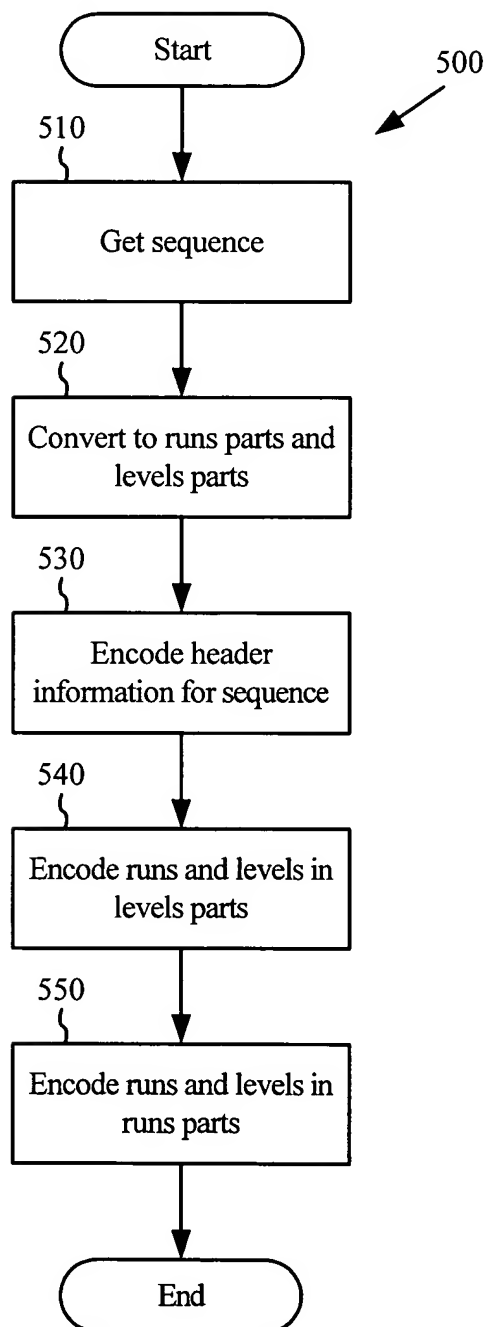


Figure 6

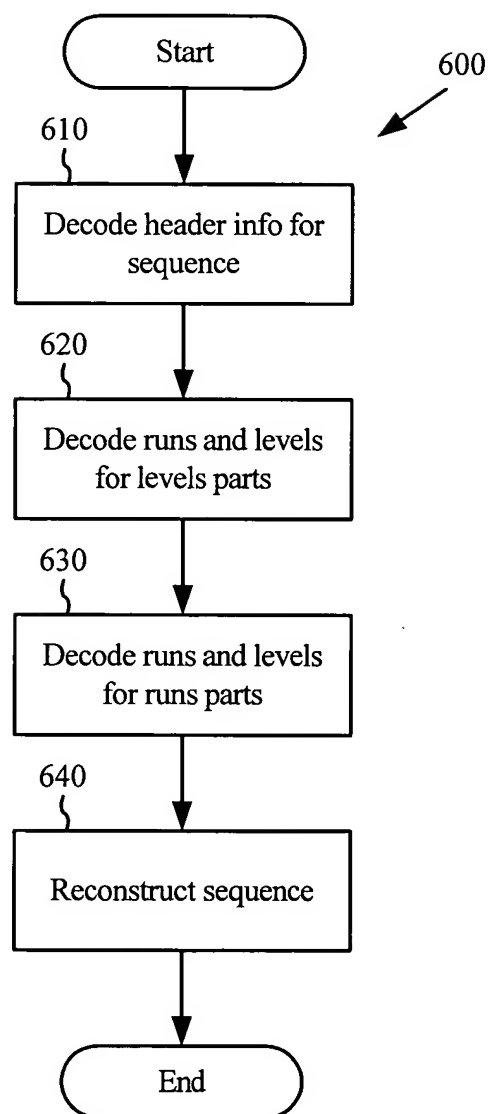


Figure 7a

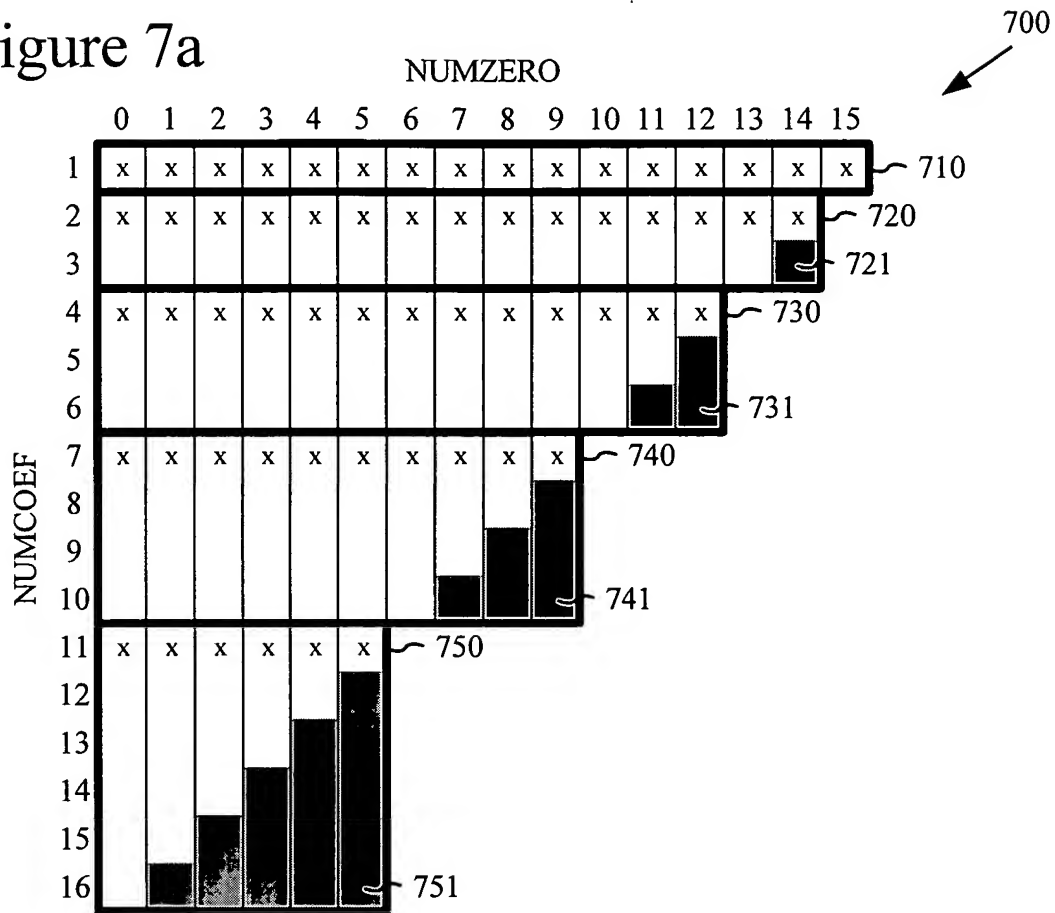


Figure 7b

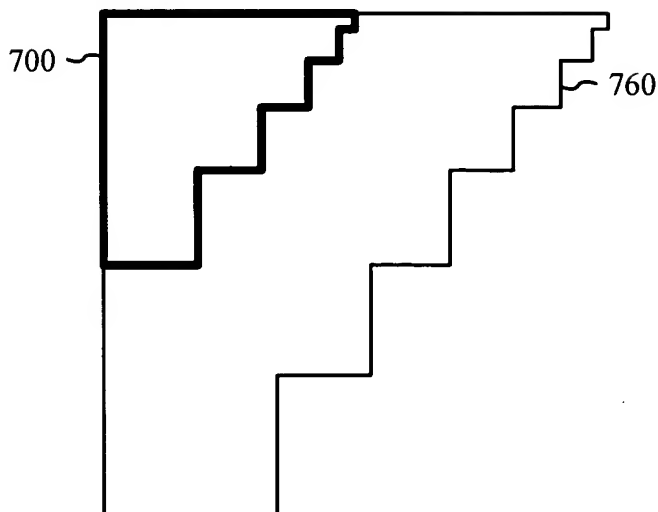


Figure 7c

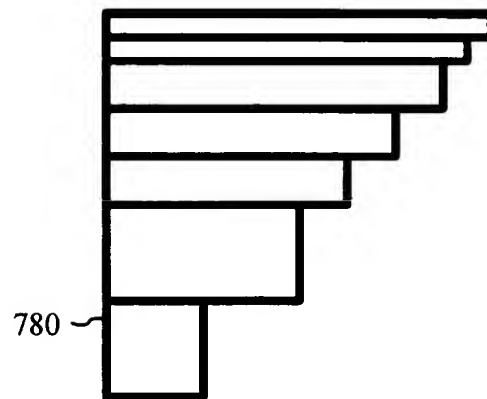


Figure 7d

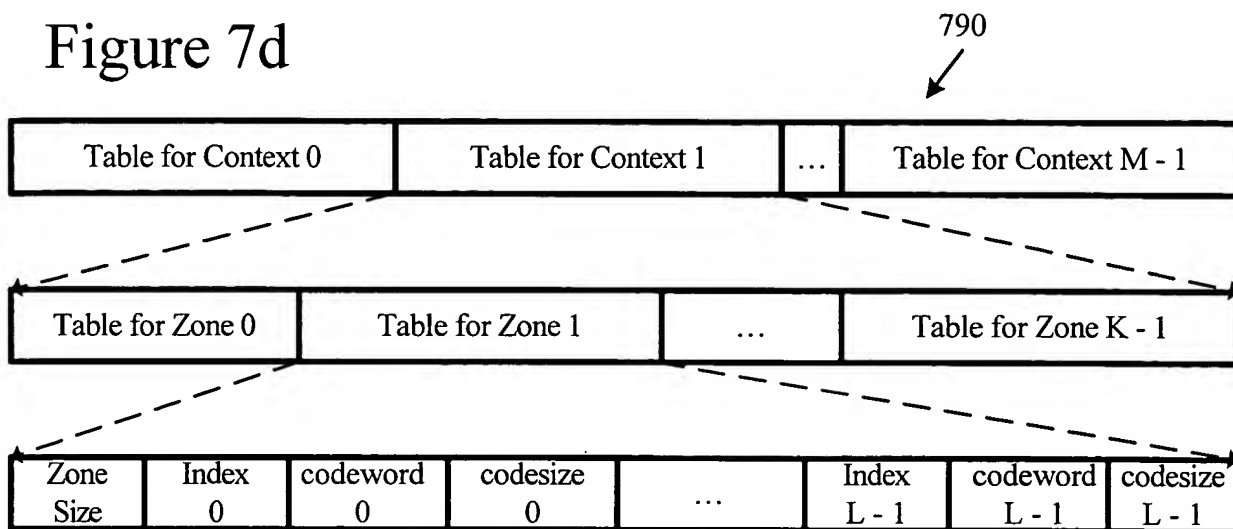


Figure 8

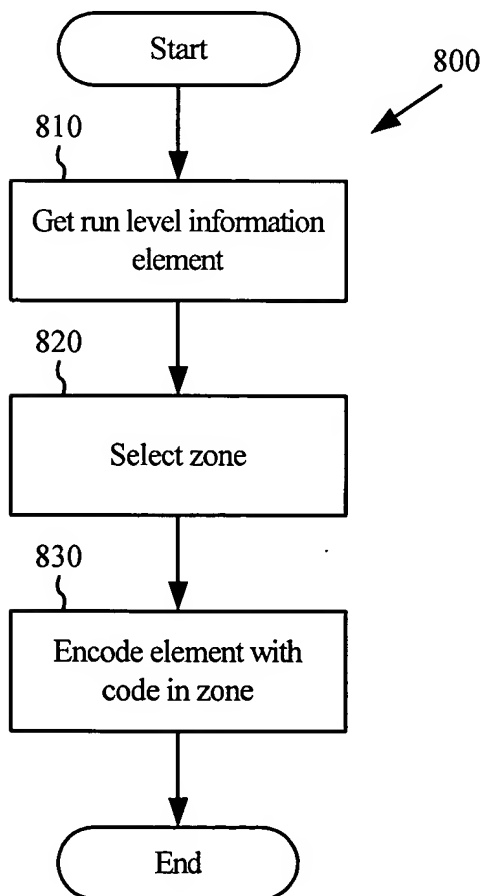


Figure 9

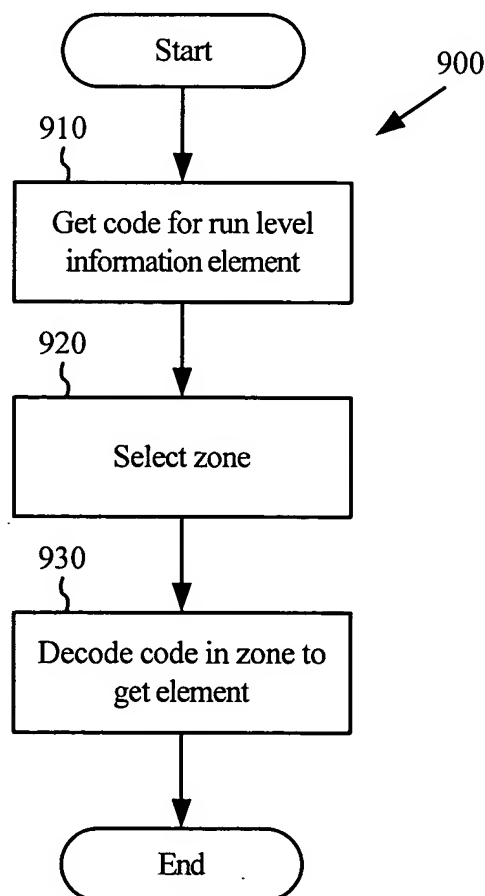


Figure 10a

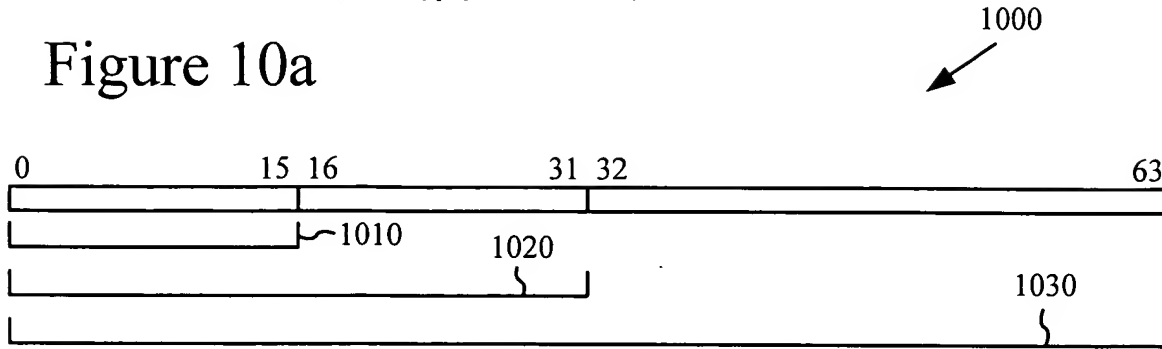


Figure 10b

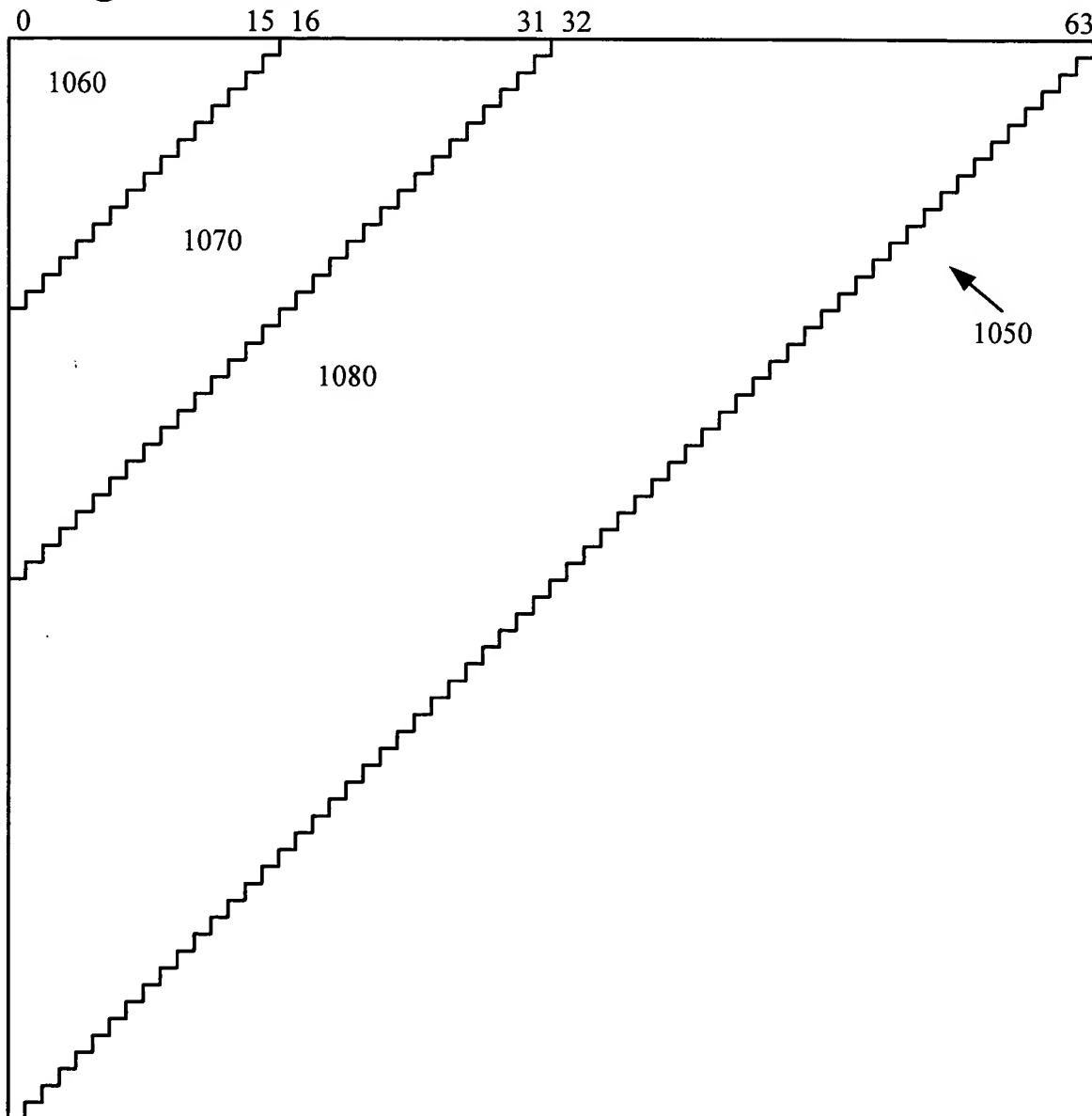


Figure 11a

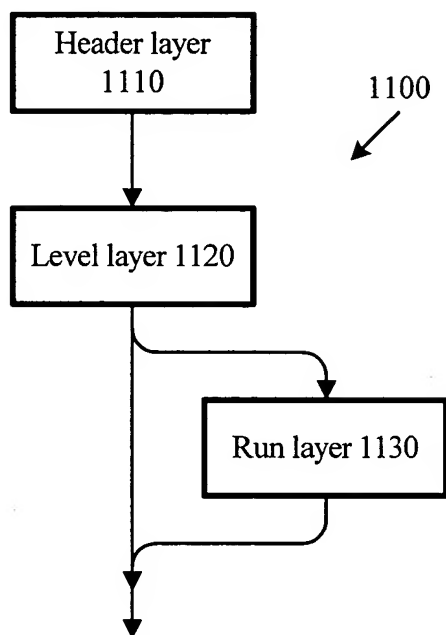


Figure 11d

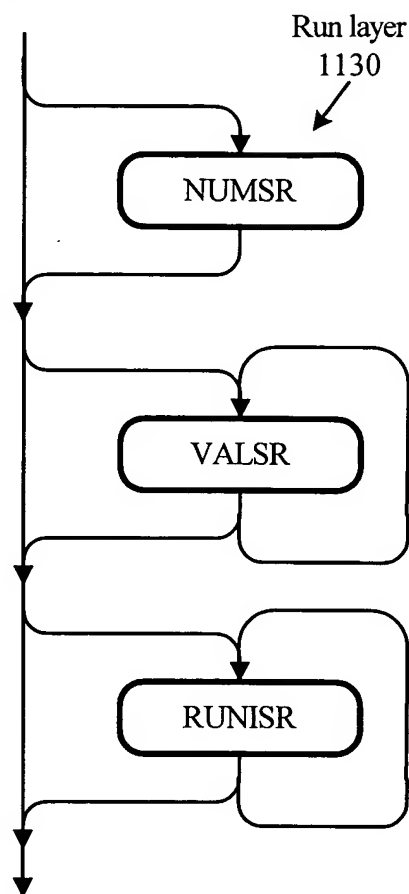


Figure 11b

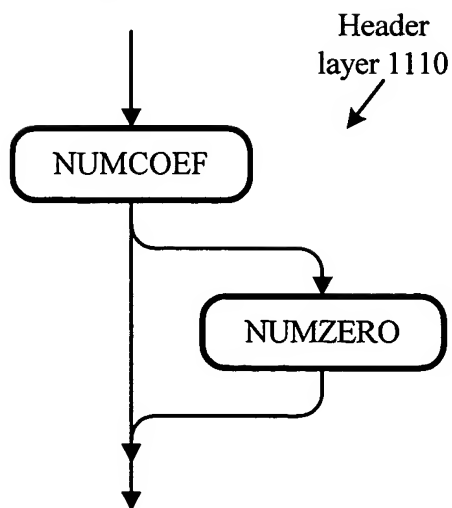




Figure 11c

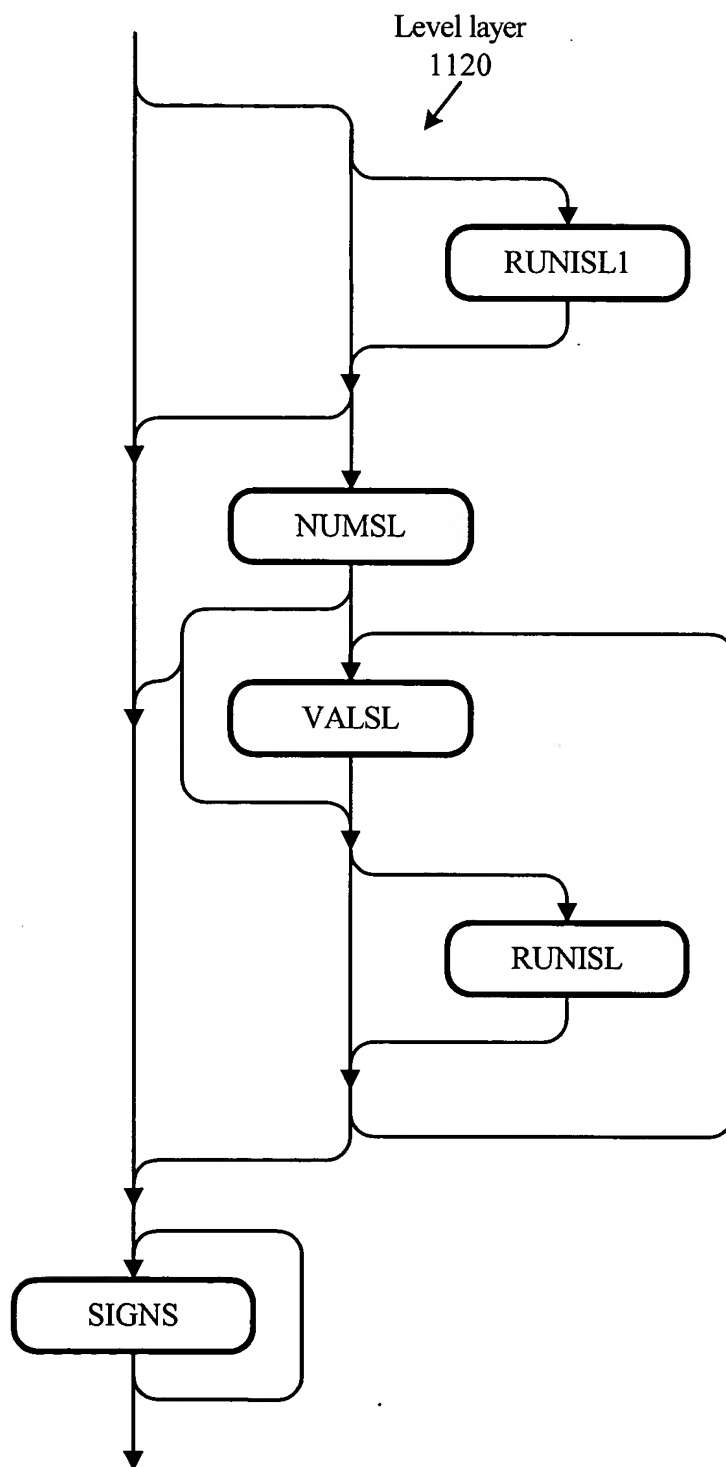


Figure 12

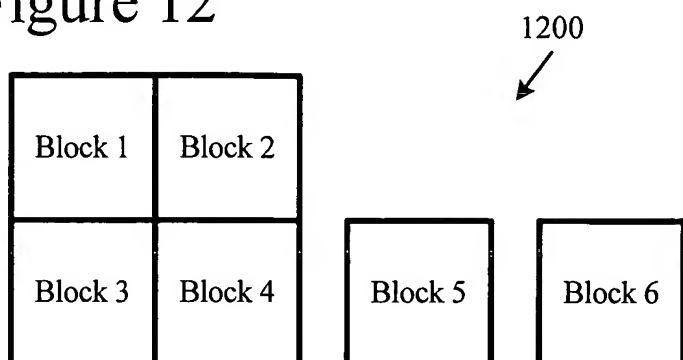


Figure 13a

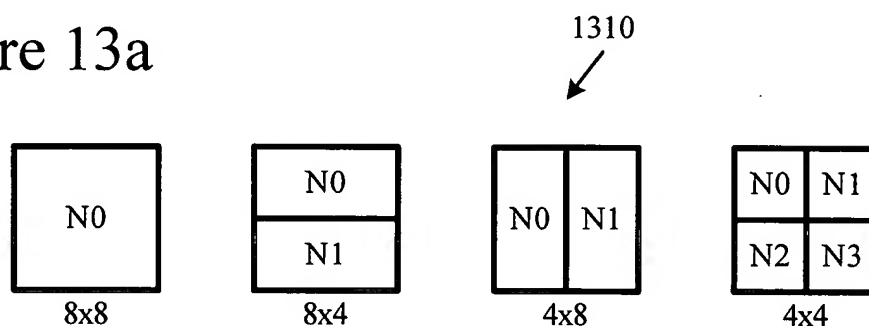


Figure 13b

1320  
↙

Current block	Neighbor in 8x8 mode	Neighbor in 8x4 mode	Neighbor in 4x8 mode	Neighbor in 4x4 mode
8x8	N0	N0 + N1	N0 + N1	N0+N1+N2+N3
8x4 top	$(N0 + 1) / 2$	N1	$(N0+N1+1) / 2$	N2 + N3
8x4 bottom	Impossible	N0	Impossible	Impossible
4x8 left	$(N0 + 1) / 2$	$(N0+N1+1) / 2$	N0	N0 + N2
4x8 right	$(N0 + 1) / 2$	$(N0+N1+1) / 2$	N1	N1 + N3
4x4 No. 0	$(N0 + 2) / 4$	$(N1 + 1) / 2$	$(N0 + 1) / 2$	N2
4x4 No. 1	$(N0 + 2) / 4$	$(N1 + 1) / 2$	$(N1 + 1) / 2$	N3
4x4 No. 2	Impossible	Impossible	Impossible	N0
4x4 No. 3	Impossible	Impossible	Impossible	N1

Figure 13c

1330



Current Sub-block	Neighbor in 8x8 mode	Neighbor in 8x4 mode	Neighbor in 4x8 mode	Neighbor in 4x4 mode
8x8	N0	N0 + N1	N0 + N1	N0+N1+N2+N3
8x4 top	$(N0 + 1) / 2$	N0	$(N0+N1+1) / 2$	N0+N1
8x4 bottom	$(N0 + 1) / 2$	N1	$(N0+N1+1) / 2$	N2+N3
4x8 left	$(N0 + 1) / 2$	$(N0+N1+1) / 2$	N1	N1+N3
4x8 right	Impossible	Impossible	N0	Impossible
4x4 No. 0	$(N0 + 2) / 4$	$(N0+1) / 2$	$(N1+1)/2$	N1
4x4 No. 1	Impossible	Impossible	Impossible	N0
4x4 No. 2	$(N0 + 2) / 4$	$(N1+1) / 2$	$(N1+1) / 2$	N3
4x4 No. 3	Impossible	Impossible	Impossible	N2

Figure 14a

1410



Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold	Context 3 Threshold	Context 4 Threshold	Context 5 Threshold
8x8	1	3	5	7	16	64
8x4/4x8	1	3	5	7	12	32
4x4	1	3	5	7	16	--
Intra	2	6	12	24	63	--

Figure 14b

1420



```
Context = 0;
While (Context < MaxContext[BlkMode] - 1 &&
      PredNumCoef > ContextThresholds_NUMCOEF[BlkMode][Context] )
    { Context ++; }
```

Figure 15

1500



Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold	Context 3 Threshold
8x8	1	4	12	64
8x4/4x8	1	4	8	32
4x4	1	3	6	16
Intra	6	18	63	--

Figure 16a

1600



```
PredNumCoef = GetPredNumCoef();
Context = GetContext_NUMCOEF(PredNumCoef);
Index = vlc_decode(HufPtr_NUMCOEF[BlkMode][Context]);
if (Index <= 3) {
    NUMCOEF = Index + 1;
    ISLONLY = TRUE; // the absolute values of all nonzero coefficients are equal to 1.
} else {
    ISLONLY = FALSE;
    if ((BlockMode is not 4x4) && (Index == 4 + BlockSize / 2) ) {
        // the last symbol in each context is escape symbol
        EscBits = (BlockMode == 8x8) ? 5 : 4;
        EscIndex = get_bits (EscBits);
        NUMCOEF = EscIndex + BlockSize / 2 + 1; }
    else { NUMCOEF = Index - 3; }
}
```

## Figure 16b

1610



```
PredNumCoef = GetPredNumCoef();
Context = GetContext_NUMCOEF_Intra (PredNumCoef);
Index = vlc_decode (HufPtr_NUMCOEF_Intra[Context] );
if (Index <= 3) {
    NUMCOEF = Index + 1;
    ISLONLY = TRUE;
} else {
    ISLONLY = FALSE;
    if ( Index == 35 ) {
        NUMCOEF = 32 + get_bits(5);
    } else { NUMCOEF = Index - 3; }
}
```

## Figure 17

1700



```
PredNumCoef = GetPredNumCoef();

Context = GetContext_NUMCOEF (PredNumCoef);

Index = vlc_decode (HufPtr_NUMCOEF[BlkMode][Context] );

if (Index >= BlockSize) {
    NUMCOEF = Index + 1 - BlockSize;
    ISLONLY = TRUE;
} else {
    NUMCOEF = Index + 1;
    ISLONLY = FALSE;
};
```

Figure 18

Block Mode	Context 0 Threshold	Context 1 Threshold
8x8	16	64
8x4/4x8	8	32
4x4	4	16

1800

Figure 19

Block Mode	Context 1 Threshold	Context 2 Threshold	Context 3 Threshold
8x8	6	16	64
8x4/4x8	4	12	32
4x4	4	8	16

1900

Figure 20a

Block Mode	Number of Zones
8x8	8
8x4/4x8	7
4x4	7
Intra 8x8	8

2000

Figure 20b

2010

```

ZoneHeight_Inter_8x8 [2][8] = {
    {1, 1, 1, 1, 4, 4, 8, 43}, // context 0
    {1, 1, 2, 4, 4, 4, 8, 39}, // context 1
};
ZoneHeight_Inter_8x4_and_4x8 [2][7] = {
    {1, 1, 1, 1, 1, 4, 22}, // context 0
    {1, 1, 1, 1, 4, 8, 15}, // context 1
};
ZoneHeight_Inter_4x4 [2][7] = {
    {1, 1, 1, 1, 1, 2, 8}, // context 0
    {1, 1, 1, 1, 1, 4, 6}, // context 1
};
ZoneHeight_Intra_8x8 [8] =
    {1, 1, 1, 1, 4, 4, 8, 42};

```

Figure 21a

2100

Block Mode	Number of Zones
8x8	8
8x4/4x8	10
4x4	15
Intra 8x8	8

Figure 21b

2110

```

ZoneHeight_Inter_8x8[3][8] = {
// 3 contexts, each has 8 zones
    {1, 1, 1, 1, 4, 8, 16, 31},
    {1, 1, 1, 1, 4, 8, 16, 31},
    {1, 1, 1, 1, 4, 8, 16, 31},
};

ZoneHeight_Inter_8x4_and_4x8[3][10] = {
    {1, 1, 1, 1, 1, 1, 1, 1, 8, 15},
    {1, 1, 1, 1, 1, 1, 1, 4, 5, 15},
    {1, 1, 1, 1, 1, 1, 2, 4, 4, 15},
};

ZoneHeight_Intra_8x8[8] =
    {1, 1, 2, 2, 4, 8, 16, 28};

```

Figure 22a

2200

```

NUMZERO_EscThresholdLeft[2][8] = {
    {0, 0, 0, 0, 0, 8, 16, 16}, // context 0
    {0, 0, 12, 12, 20, 20, 16, 16}, // context 1
};

NUMZERO_EscThresholdRight[2][8] = {
    {32, 32, 32, 32, 32, 40, 48, 44}, // context 0
    {32, 32, 44, 44, 52, 52, 48, 40}, // context 1
};

NUMZERO_EscThresholdLeft_Intra[8] = {0, 0, 0, 0, 0, 4, 12, 16};
NUMZERO_EscThresholdRight_Intra[8] = {32, 32, 32, 32, 32, 36, 44, 43};

```

## Figure 22b

```
PredNumZero = GetPredNumZero();
Context = GetContext_NUMZERO(PredNumZero);
Zone = GetZone_NUMZERO(NUMCOEF);
ZoneHead = GetZoneHead_NUMZERO(NUMCOEF);
RightShift = NUMCOEF - ZoneHead;

index = vlc_decode (HufPtr_NUMZERO[BlkMode][Context][Zone]);
if (block mode is not 8x8) {
    NUMZERO = index - RightShift;
} else {
    // check escape symbol
    if (index > 0) {
        NUMZERO = index - 1 +
            NUMZERO_EscThresholdLeft[Context][Zone] - RightShift;
    } else {
        EscIndex = get_bits (5);
        if (EscIndex < NUMZERO_EscThresholdLeft[Context][Zone]) {
            // left margin
            NUMZERO = EscIndex - RightShift;
        } else {
            // right margin
            NUMZERO = EscIndex -
                NUMZERO_EscThresholdLeft[Context][Zone] +
                NUMZERO_EscThresholdRight[Context][Zone] - RightShift;
        }
    }
}
```

2210  
↙



## Figure 22c

2220



```
Zone = GetZone_NUMZERO_Intra(NUMCOEF);
ZoneHead = GetZoneHead_NUMZERO_Intra(NUMCOEF);
RightShift = NUMCOEF - ZoneHead;

index = vlc_decode(HufPtr_NUMZERO_Intra[Zone]);

if (index > 0) {
    NUMZERO = index - 1 + NUMZERO_EscThresholdLeft_Intra[Zone] - RightShift;
} else {
    EscIndex = get_bits(5);
    if (EscIndex < NUMZERO_EscThresholdLeft_Intra[Zone]) {
        // left margin
        NUMZERO = EscIndex - RightShift;
    } else {
        // right margin
        NUMZERO = EscIndex - NUMZERO_EscThresholdLeft_Intra[Zone] +
            NUMZERO_EscThresholdRight_Intra[Zone] - RightShift;
    }
}
```

## Figure 23

2300



```
PredNumZero = GetPredNumZero();

Context = 0;
If (BlkMode == INTER) {
    Context = GetContext_NUMZERO(PredNumZero);
}

Zone = GetZone_NUMZERO(NUMCOEF);

NUMZERO = vlc_decode(HufPtr_NUMZERO[BlkMode][Context][Zone]);
```

## Figure 24

2400  
↙

```

RUNISL1 = Decode_RUNISL1( );
if (NUMCOEF - RUNISL1 > 0) {
    //Function returns NUMSL and SingleTwoFound.
    Decode_NUMSL(&NUMSL, &SingleTwoFound);
    ISLLeft = NUMCOEF - RUNISL1 - NUMSL;

    if (SingleTwoFound == FALSE) {
        LevelZone = 0;
        LevelThreshold[BlkMode] = IniLevelThreshold[BlkMode];
        ShiftLevel = (NUMSL == 1);
        for (n = NUMSL - 1; n >= 0; n--) {
            VALSL(n) = Decode_VALSL( );
            if (n > 0 && ISLLeft) {
                RUNISL(n) = Decode_RUNISL( );
                ISLLeft = ISLLeft - RUNISL(n);
            }
        }
    }
}
Decode_Signs( );

```

## Figure 25a

2500  
↙

NUMCOEF	RUNISL1=0	RUNISL1 = 1	RUNISL1 = 2	RUNISL1 = 3
2	0	1	--	--
3	10	0	11	--
4	00	01	10	11

## Figure 25b

2510



```
If (ISLONLY == TRUE) {  
    RUNISL1 = NUMCOEF;  
} else {  
    If (NUMCOEF == 1) {  
        RUNISL1 = 0;  
    } else {  
        If (NUMCOEF <= 4) {  
            RUNISL1 = Vlc_decode ( HufPtr_RUNISL1_1[ NUMCOEF - 2] );  
        } else {  
            Zone = GetZone_RUNISL1( NUMCOEF);  
            index = vlc_decode ( HufPtr_RUNISL1_2 [Zone]);  
            if ( Zone > 6 && index == 15 ) {  
                RUNISL1= 15 + get_bits( 6 );  
            } else {  
                RUNISL1= index;  
            }  
        }  
    }  
}
```

## Figure 25c

2520



```
If (ISLONLY == TRUE) {  
    RUNISL1 = NUMCOEF;  
} else {  
    If (NUMCOEF == 1) {  
        RUNISL1 = 0;  
    } else {  
        If (NUMCOEF <= 4) {  
            RUNISL1 = Vlc_decode(HufPtr_RUNISL1_1[NUMCOEF - 2]);  
        } else {  
            Zone = GetZone_RUNISL1_Intra(NUMCOEF);  
            index = Vlc_decode(HufPtr_RUNISL1_2_Intra[Zone]);  
            if (index < 33) {  
                RUNISL1 = index;  
            } else {  
                RUNISL1 = 33 + get_bits(5);  
            }  
        }  
    }  
}
```

Figure 26a

2600



```
If (NUMCOEF > 1) {
    If (NUMCOEF <= 4) {
        Index = Vlc_decode(HufPtr_RUNISL1_1[NUMCOEF - 2]);
    } else {
        Zone = GetZone_RUNISL1( NUMCOEF);

        RUNISL1=Vlc_decode(HufPtr_RUNISL1_2[NUMCOEF_Context][Zone]);
    }
}
```

Figure 26b

2610



```
If (NUMCOEF > 1) {
    If (NUMCOEF <= 4) {
        Index = Vlc_decode(HufPtr_RUNISL1_1_Intra[NUMCOEF - 2]);
    } else {
        Zone = GetZone_RUNISL1_Intra( NUMCOEF);
        RUNISL1 = Vlc_decode(HufPtr_RUNISL1_2_Intra[Zone]);
    }
}
```

Figure 26c

2620



NUMCOEF	RUNISL1=0	RUNISL1 = 1	RUNISL1 = 2	RUNISL1 = 3
2	0	1	--	--
3	10	0	11	--
4	00	01	10	11

Figure 26d

NUMCOEF	RUNISL1=0	RUNISL1 = 1	RUNISL1 = 2	RUNISL1 = 3
2	0	1	--	--
3	10	11	0	--
4	00	01	10	11

2630

Figure 27a

Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold
8x8	10	30	64
8x4/4x8	4	16	32
4x4	4	9	16

2700

Figure 27b

```

ZoneHeight_NUMSL[3][8] =
{
    {1, 1, 1, 1, 1, 1, 1, 57},
    {1, 1, 1, 1, 2, 4, 8, 46},
    {4, 4, 2, 2, 4, 4, 8, 36},
};

```

2710

Figure 27c

2720



```
Context = GetContext_Level(NUMCOEF);
Zone = GetZone_NUMSL(NUMCOEF - RUNISL1);
index = vlc_decode (HufPtr_NUMSL[Context][Zone]);
if (index == 0) {
    SingleTwoFound = TRUE;
    NUMSL = 1;
} else {
    SingleTwoFound = FALSE;
    if (index < 33) {
        NUMSL = index;
    } else {
        NUMSL = 33 + get_bits(5);
    }
}
```

Figure 27d

2730



```
Zone = GetZone_NUMSL_Intra(NUMCOEF - RUNISL1);
index = vlc_decode (HufPtr_NUMSL_Intra[Zone]);
if (index == 0) {
    NUMSL = 1;
    SingleTwoFound = TRUE;
} else {
    NUMSL = index;
    SingleTwoFound = FALSE;
}
```

## Figure 28

2800



```
index = vlc_decode(HufPtr_VALSL[LevelZone]);
if (index < 30) {
    VALSL = index + 2;
} else {
    EscScale = 1;
    while (!get_bits(1)) {
        EscScale ++;
    }
    VALSL = EscScale * 32 + get_bits(5);
}
if (ShiftLevel == TRUE) {
    VALSL ++;
    ShiftLevel = FALSE;
}
if (VALSL > LevelThreshold[BlkMode] && LevelZone < 3) {
    LevelThreshold[BlkMode] = LevelThreshold[BlkMode] * 2;
    LevelZone = LevelZone + 1;
}
```



## Figure 29a

2900



```
Zone = GetZone_RUNISL(ISLLeft);  
index = vlc_decode(HufPtr_RUNISL[Context][Zone]);  
if (ISLLeft >= 32 && index == 32) {  
    RUNISL = 32 + get_bits(5);  
} else {  
    RUNISL = index;  
}
```

## Figure 29b

2910



```
Zone = GetZone_RUNISL_Intra(ISLLeft);  
index = vlc_decode(HufPtr_RUNISL_Intra[Zone]);  
if (ISLLeft >= 32 && index == 32) {  
    RUNISL = 32 + get_bits(5);  
} else {  
    RUNISL = index;  
}
```

## Figure 30

3000



```
if (NUMZERO == 1) {  
    NUMSR = 1;  
} else {  
    NUMSR = Decod_NUMSR();  
    Decode_VALSR();  
}  
ISRLeft = NUMCOEF - NUMSR;  
if (ISRLeft) {  
    Decode_RUNISR();  
}
```

Figure 31a

Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold
8x8	20	32	64
8x4/4x8	10	16	32
4x4	8	12	16

3100



Figure 31b

```
Context_NUMSR = GetContext_NUMSR (NUMZERO);
MaxNUMSR = min(NUMCOEF, NUMZERO);
Zone = GetZone_NUMSR(MaxNUMSR);
Index = vlc_decode(HufPtr_NUMSR[Context_NUMSR][Zone]);
NUMSR = index + 1;
```

3110



Figure 31c

```
Context_NUMSR = GetContext_NUMSR_Intra(NUMZERO);
MaxNUMSR = min(NUMCOEF, NUMZERO);
Zone = GetZone_NUMSR_Intra(MaxNUMSR);
Index = vlc_decode(HufPtr_NUMSR_Intra[Context_NUMSR][Zone]);
NUMSR = index + 1;
```

3120



Figure 32a

Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold
8x8	8	20	64
8x4/4x8	6	12	32
4x4	4	10	16

3200

Figure 32b

```

Context_VALSR = GetContext_VALSR(NUMCOEF);
SRSumLeft = NUMZERO;
MaxVALSR = SRSumLeft - NUMSR + 1;
for (SRLeft = NUMSR; SRLeft > 0; SRLeft --) {
    if (SRLeft == 1) {
        VALSR[0] = SRSumLeft;
        break;
    } else if (SRLeft == SRSumLeft) {
        // set all remaining SRs to 1.
        break;
    }
    Zone = GetZone_VALSR(MaxVALSR);
    Index = vlc_decode(HufPtr_VALSR[Context_VALSR][Zone]);
    if (MaxVALSR >= 33 && Index == 32) {
        VALSR[SRLeft - 1] = 33 + get_bits(5);
    } else {
        VALSR[SRLeft - 1] = Index + 1;
    }
    SRSumLeft = SRSumLeft - VALSR[SRLeft - 1];
    MaxVALSR = MaxVALSR - VALSR[SRLeft - 1] + 1;
}

```

3210

## Figure 33

3300



```
Context_VALSR = GetContext_VALSR(NUMZERO);
MaxVALSR = SRSumLeft - NUMSR + 1;
for (SRLeft = NUMSR; SRLeft > 0; SRLeft--) {
    if (SRLeft == 1) {
        VALSR = SRSumLeft;
        // save VALSR
        break;
    } else if (SRLeft == SRSumLeft) {
        // set all remaining SRs to 1.
        break;
    }
    Zone = GetZone_VALSR(MaxVALSR);
    Index = vlc_decode(HufPtr_VALSR[Context_VALSR][Zone]);
    VALSR = index + 1;
    // save VALSR
    SRSumLeft = SRSumLeft - VALSR;
    MaxVALSR = MaxVALSR - VALSR + 1;
}
```

## Figure 34a

3400



Block Mode	Context 0 Threshold	Context 1 Threshold	Context 2 Threshold
8x8	8	20	63
8x4/4x8	6	12	31
4x4	4	10	15

## Figure 34b

3410



```
ISRLeft = NUMCOEF - NUMSR;  
if (ISRLeft) {  
    Context = getContext_RUNISR(NUMZERO);  
    for (n = 0; n < NUMSR && ISRLeft > 0; n++) {  
        Zone = GetZone_RUNISR(ISRLeft);  
        Index = vlc_decode(HufPtr_RUNISR[Context][Zone]);  
        if (ISRLeft >= 32 && Index == 32) {  
            RUNISR[n] = 32 + get_bits(5);  
        } else {  
            RUNISR[n] = Index;  
        }  
        ISRLeft = ISRLeft - RUNISR[n];  
    }  
}
```

## Figure 35

3500



```
ISRLeft = NUMCOEF - NUMSR;  
if (ISRLeft) {  
    Context = getContext_RUNISR(NUMZERO);  
    for (n = 0; n < NUMSR && ISRLeft > 0; n++) {  
        Zone = GetZone_RUNISR(ISRLeft);  
        RUNISR[n] = vlc_decode(HufPtr_RUNISR[Context][Zone]);  
        ISRLeft = ISRLeft - RUNISR[n];  
    }  
}
```

Figure  
36a

0	2	3	9	10	21	22	36
1	4	8	11	20	23	35	37
5	7	12	19	24	34	38	49
6	13	18	25	33	39	48	50
14	16	26	32	40	47	51	58
15	27	31	41	46	52	57	59
17	29	42	44	53	55	60	62
28	30	43	45	54	56	61	63

3601



Normal Intra Block Scan Pattern

Figure  
36b

0	1	3	4	10	11	22	23
2	5	9	12	21	24	36	37
6	8	13	20	25	35	38	48
7	14	19	26	34	39	47	49
15	18	27	33	40	46	50	57
16	28	32	41	45	51	56	58
17	30	42	44	52	55	59	62
29	31	43	53	54	60	61	63

3602



Horizontal Intra Block Scan Pattern

Figure  
36c

0	3	8	9	20	21	34	35
1	7	10	19	22	33	36	49
2	11	18	23	32	37	48	50
4	12	17	24	31	38	47	51
5	16	25	30	39	46	52	57
6	15	29	40	45	53	56	58
13	26	28	41	44	55	59	62
14	27	42	43	54	60	61	63

3603



Vertical Intra Block Scan Pattern

Figure 36d

3611

0	2	3	9	10	23	24	38
1	4	8	11	22	25	37	39
5	7	12	21	26	36	40	51
6	13	20	27	35	41	50	52
14	19	28	34	42	49	53	60
15	18	33	43	48	54	59	61
16	29	32	44	47	55	58	62
17	30	31	45	46	56	57	63

8x8 Inter Block Scan Pattern for Progressive Content

Figure 36e

3612

0	2	4	7	10	14	21	27
1	5	6	11	13	17	24	29
3	9	12	15	18	22	25	30
8	16	19	20	23	26	28	31

8x4 Inter Block Scan Pattern for Progressive Content

Figure 36f

3613

0	1	3	13
2	4	8	17
5	6	11	24
7	10	15	26
9	14	20	28
12	19	23	29
16	21	25	30
18	22	27	31

4x8 Inter Block Scan Pattern for  
Progressive Content

Figure 36g

3614

0	3	7	11
1	4	8	12
2	6	9	14
5	10	13	15

4x4 Inter Block Scan Pattern  
for Progressive Content

Figure 36h

0	2	6	13	17	29	33	38
1	5	12	16	28	32	37	39
3	11	15	27	31	36	40	51
4	14	22	30	35	41	50	52
7	18	23	34	42	49	53	60
8	19	24	43	48	54	59	61
9	20	25	44	47	55	58	62
10	21	26	45	46	56	57	63

3621



8x8 Inter Block Scan Pattern for Interlaced Content

Figure 36i

0	4	6	10	13	17	21	27
1	5	9	14	16	18	24	29
2	7	11	15	19	22	25	30
3	8	12	20	23	26	28	31

3622



8x4 Inter Block Scan Pattern for Interlaced Content

Figure 36j

0	1	2	9
3	5	8	22
4	7	15	24
6	14	17	26
10	16	19	28
11	18	23	29
12	20	25	30
13	21	27	31

3623



4x8 Inter Block Scan Pattern for  
Interlaced Content

Figure 36k

0	4	7	11
1	5	9	13
2	6	10	14
3	8	12	15

3624



4x4 Inter Block Scan Pattern  
for Interlaced Content